Affine Equivariant Networks Based on Differential Invariants

Supplementary Material

A. Proof of Theorem 7

Proof. Since $\mathcal{F}'$ is a set of bounded functions, the norm $\|\cdot\|_{\sup}$ is well-defined. $\forall \mathbf{u} \in \mathcal{F}, g \in G$, we have

$$\begin{aligned}
\|\boldsymbol{\mathcal{J}}(\cdot, g\cdot\mathbf{u})\|_{\sup} &= \sup_{\mathbf{x}\in X} \|\boldsymbol{\mathcal{J}}(\mathbf{x}, g\cdot\mathbf{u})\|_{\infty} \\
&= \sup_{\mathbf{x}\in X} \|w(g)\boldsymbol{\mathcal{J}}(g^{-1}\cdot\mathbf{x}, \mathbf{u})\|_{\infty} \\
&= w(g) \sup_{\mathbf{x}\in X} \|\boldsymbol{\mathcal{J}}(g^{-1}\cdot\mathbf{x}, \mathbf{u})\|_{\infty} \qquad (14) \\
&= w(g) \sup_{\mathbf{x}\in X} \|\boldsymbol{\mathcal{J}}(\mathbf{x}, \mathbf{u})\|_{\infty} \\
&= w(g)\|\boldsymbol{\mathcal{J}}(\cdot, \mathbf{u})\|_{\sup}.
\end{aligned}$$

Then we apply the above property of the norm $\|\cdot\|_{\sup}$ to complete the proof. $\forall \mathbf{u} \in \mathcal{F}, g \in G, \mathbf{x} \in X$, we have

$$\begin{aligned}
\boldsymbol{\mathcal{I}}(g\cdot\mathbf{x}, g\cdot\mathbf{u}) &= \frac{1}{\|\boldsymbol{\mathcal{J}}(\cdot, g\cdot\mathbf{u})\|_{\sup}} \boldsymbol{\mathcal{J}}(g\cdot\mathbf{x}, g\cdot\mathbf{u}) \\
&= \frac{1}{w(g)\|\boldsymbol{\mathcal{J}}(\cdot, \mathbf{u})\|_{\sup}} w(g)\boldsymbol{\mathcal{J}}(\mathbf{x}, \mathbf{u}) \qquad (15) \\
&= \frac{1}{\|\boldsymbol{\mathcal{J}}(\cdot, \mathbf{u})\|_{\sup}} \boldsymbol{\mathcal{J}}(\mathbf{x}, \mathbf{u}) \\
&= \boldsymbol{\mathcal{I}}(\mathbf{x}, \mathbf{u}).
\end{aligned}$$

Therefore, $\boldsymbol{\mathcal{I}}$ is a k-dimensional invariant of G. □

B. Polynomial relative differential invariants

In this work, we implement equivariant networks for three non-Euclidean groups: the scale group, the rotation-scale group, and the affine group. Targeting each group, we first derive its polynomial relative differential invariants up to order 2, and then compute SupNorm normalized differential invariants to build the equivariant models. Elements in the affine group and its subgroups can be represented as $g = (\mathbf{A}, \mathbf{b})$, where $\mathbf{A} \in \mathbb{R}^{2\times 2}$ is invertible and $\mathbf{b} \in \mathbb{R}^2$. For the scale group, $\mathbf{A} = \begin{pmatrix} C_S & 0 \\ 0 & C_S \end{pmatrix}, C_S > 0$; for the rotation-scale group, $\mathbf{A} = C_{RS} \cdot \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, C_{RS} > 0, \theta \in [0, 2\pi)$; for the affine group, we consider $\mathbf{A} = C_A \cdot \tilde{\mathbf{A}}, C_A > 0, \det(\tilde{\mathbf{A}}) = 1$. Polynomial relative differential invariants involved in our models for these groups are shown in Table 5.

C. Implementation details of models

In this section, we give a detailed description of implementation of InvarPDEs and InvarLayer.

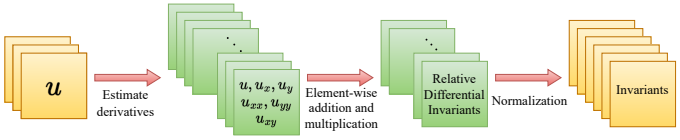

Figure 4. The implementation process of computing SupNorm normalized differential invariants (SNDI).

To start with, we introduce the implementation process of computing SupNorm normalized differential invariants (SNDI) (see Figure 4). Firstly, we estimate derivatives by applying derivatives of a Gaussian kernel. We set the standard deviation σ of the Gaussian kernel to 0.99 and choose the kernel size 7. Secondly, we compute polynomial relative differential invariants through element-wise addition and multiplication of the derivatives. Finally, we normalize polynomial relative differential invariants with the same weight and degree together to obtain SNDIs, where the computation of SupNorm is implemented by applying Max-Pooling over those channels that are to be normalized together.

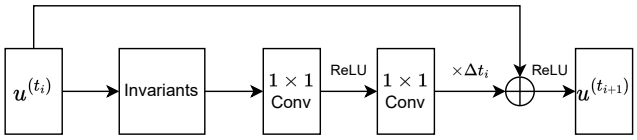

Figure 5. The architecture for each layer of InvarPDEs-Net.

InvarPDEs-Net consists of iterative processes of multiple symmetric PDEs. Each iteration of one evolutionary PDE can be viewed as a layer of the network, shown in Figure 5. In each layer, we first compute invariants of the input $\mathbf{u}^{(t_i)}$ of this layer, and then perform Conv-ReLU-Conv (denoting 1×1 convolutions as "Conv"). The result is multiplied by Δt_i and added to $\mathbf{u}^{(t_i)}$, where we set Δt_i as a learnable parameter. After passing through ReLU, we obtain the output $\mathbf{u}^{(t_{i+1})}$ of this layer. The iterations of the same PDE are stacked together, and we connect iterations of different PDEs through Conv-ReLU, where the input and output dimensions of Conv are determined by the dimensions of the previous and next PDEs, respectively. In our implementation, we use hyperparameter *channel* to specify the dimension of the first PDE, and the dimension of each subsequent PDE is twice that of the previous one. We use the hyperparameter *iteration* to specify the total number of iterations of all PDEs, namely the network depth. In each layer, the input dimension of the first Conv is the same as

Relative differential invariants of the scale group	Weight	Degree
$u^{[k]}, \quad (1 \leq k \leq n)$	1	1
$u_x^{[k]}, \quad (1 \leq k \leq n)$	$1/C_S$	1
$u_y^{[k]}, \quad (1 \leq k \leq n)$	$1/C_S$	1
$u_{xx}^{[k]}, \quad (1 \leq k \leq n)$	$1/C_S^2$	1
$u_{yy}^{[k]}, \quad (1 \leq k \leq n)$	$1/C_S^2$	1
$u_{xy}^{[k]}, \quad (1 \leq k \leq n)$	$1/C_S^2$	1
Relative differential invariants of the rotation-scale group	**Weight**	**Degree**
$u^{[k]}, \quad (1 \leq k \leq n)$	1	1
$(u_x^{[k]})^2 + (u_y^{[k]})^2, \quad (1 \leq k \leq n)$	$1/C_{RS}^2$	2
$u_x^{[k+1]}u_x^{[k]} + u_y^{[k+1]}u_y^{[k]}, \quad (1 \leq k \leq n-1)$	$1/C_{RS}^2$	2
$(u_x^{[k]})^2 u_{xx}^{[k]} + 2u_x^{[k]}u_y^{[k]}u_{xy}^{[k]} + (u_y^{[k]})^2 u_{yy}^{[k]}, \quad (1 \leq k \leq n)$	$1/C_{RS}^4$	3
$(u_y^{[k]})^2 u_{xx}^{[k]} - 2u_x^{[k]}u_y^{[k]}u_{xy}^{[k]} + (u_x^{[k]})^2 u_{yy}^{[k]}, \quad (1 \leq k \leq n)$	$1/C_{RS}^4$	3
$u_x^{[k]}u_y^{[k]}(u_{yy}^{[k]} - u_{xx}^{[k]}) + u_{xy}^{[k]}\left((u_x^{[k]})^2 - (u_y^{[k]})^2\right) \quad (1 \leq k \leq n)$	$1/C_{RS}^4$	3
Relative differential invariants of the affine group	**Weight**	**Degree**
$u^{[k]}, \quad (1 \leq k \leq n)$	1	1
$u_x^{[k+1]}u_y^{[k]} - u_x^{[k]}u_y^{[k+1]}, \quad (1 \leq k \leq n-1)$	$1/C_A^2$	2
$u_{xx}^{[k]}u_{yy}^{[k]} - (u_{xy}^{[k]})^2, \quad (1 \leq k \leq n)$	$1/C_A^4$	2
$(u_y^{[k]})^2 u_{xx}^{[k]} - 2u_x^{[k]}u_y^{[k]}u_{xy}^{[k]} + (u_x^{[k]})^2 u_{yy}^{[k]}, \quad (1 \leq k \leq n)$	$1/C_A^4$	3
$u_y^{[k+1]}u_y^{[k]}u_{xx}^{[k]} - (u_x^{[k]}u_y^{[k+1]} + u_x^{[k+1]}u_y^{[k]})u_{xy}^{[k]} + u_x^{[k+1]}u_x^{[k]}u_{yy}^{[k]}, \quad (1 \leq k \leq n-1)$	$1/C_A^4$	3
$u_y^{[k+1]}u_y^{[k]}u_{xx}^{[k+1]} - (u_x^{[k]}u_y^{[k+1]} + u_x^{[k+1]}u_y^{[k]})u_{xy}^{[k+1]} + u_x^{[k+1]}u_x^{[k]}u_{yy}^{[k+1]}, \quad (1 \leq k \leq n-1)$	$1/C_A^4$	3

Table 5. We present relative differential invariants of the scale group, the rotation-scale group and the affine group for vector functions $\mathbf{u} \triangleq (u^{[1]}, u^{[2]}, ..., u^{[n]})^\top$ involved in our models.

the number of invariants, and the output dimension is specified by the hyperparameter $hidden\_channel$. The input dimension of the second Conv is $hidden\_channel$, and the output dimension is the same as the dimension of the PDE. Batch normalization (BN) is applied after each Conv. For the final output feature $\mathbf{u}^{(t_N)}$, we perform MP-FC-ReLU-FC (denoting global spatial Max-Pooling as "MP" and fully connected layers as "FC") to obtain the classification result, where the output dimension of the first FC is fixed at 64.

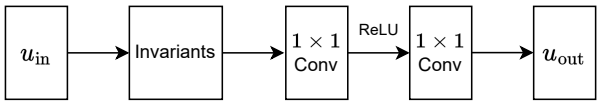

Figure 6. The architecture for InvarLayer.

InvarLayer is an equivariant layer extracted and adapted from the iterative process of a symmetric PDE, which allows for free specification of input and output channel numbers, as illustrated in Figure 6. It can serve as a drop-in replacement for convolutional networks of various architectures. The layer first computes invariants of the input $\mathbf{u}_{in}$, and then performs Conv-ReLU-Conv to get the output $\mathbf{u}_{out}$ of the layer. The dimension of $\mathbf{u}_{in}$ is specified by the hyperparameter $channel\_in$. The input dimension of the first Conv is the same as the number of invariants. Different from InvarPDEs-Net, the output dimension of the second Conv is not restricted but is specified by the hyperparameter $channel\_out$, indicating the dimension of $\mathbf{u}_{out}$. We fix the input dimension of the second Conv as $channel\_out$ to reduce the number of hyperparameters. We do not include additional BN in InvarLayer, since usually there are already BN after convolutional layers in most convolutional network structures. Given a convolutional network, we replace every convolutional layer with InvarLayer by choosing appropriate values for $channel\_in$ and $channel\_out$. If the convolutional layer has a stride more than 1, we add a Max-Pooling with an appropriate kernel size before the second Conv in InvarLayer.

D. Analysis of computational complexity

We mainly focus on the computational complexity of computing invariants. As shown in Figure 4, the process consists of three steps. The first step involves derivatives estimation using Gaussian kernel convolutions, where both time and memory usage grow linearly with the input size. The second step involves element-wise addition and multiplication, with time and space complexity also linearly dependent on the input size. The third step, normalization,

Input Size	32×32	64×64	128×128	256×256	512×512	1024×1024	2048×2048
Memory (MB)	4.16	16.63	66.50	266.00	1064.00	4256.00	17024.00
FLOPs	5.88×10^7	2.35×10^8	9.41×10^8	3.76×10^9	1.51×10^{10}	6.02×10^{10}	240.79×10^{11}
Time (ms)	27.39	28.29	29.95	31.30	93.28	361.01	1447.20

Table 6. The memory usage required for model inference, FLOPs, and inference time per image.

is implemented by Max-Pooling, where the computational complexity is proportional to the input size. Besides, after computing invariants, subsequent 1×1 Conv operations in both InvarPDEs-Net and InvarLayer also exhibit computational complexity linearly related to the input size. As a result, the time and space complexity during model inference grow linearly with the input size. We measure computational complexity of our InvarLayer used in Section 4.3, and report the numerical results in Table 6. We utilize *torchstat* to measure the memory usage required for model inference and the FLOPs, and conduct explicit timing measurements during model inference. As shown in Table 6, both the memory usage and the FLOPs exhibit linear growth with the input size. For large input sizes, the explicit inference time is almost proportional to the input size, while it grows slowly with small input sizes. This behavior may be attributed to the underlying parallelism and optimizations in low-level computations.

E. Additional experimental results

Mentioned in Section 4.3, we additionally go back to the conventional setting, training on affNIST and testing on affNIST. The results are shown in Table 7. Compared with results of other models under the same setting, InvarLayer performs the best again.

Models	Accuracy	Parameters
ITN [77]	98.91	–
DE-CNNs [73]	99.08	> 2.5M
InvarPDEs-Net (Ours)	97.13 ± 0.05	373K
InvarLayer (Ours)	$\mathbf{99.25 \pm 0.08}$	365K

Table 7. Test accuracy (%) on affNIST after training on affNIST.

F. Details of experiments

For empirical validation, we implement InvarPDEs-Net and InvarLayer for three non-Euclidean groups: the scale group, the rotation-scale group, and the affine group. We conduct classification experiments on image datasets with different group transformations. In all experiments, we do not use data augmentation to emphasize the innate equivariance of networks. Models are trained using AdamW with a batch size of 32 and the CosineLR schedule on a single RTX 3090. We report the mean test performances of models, taken at the final epoch, over 6 independent training runs.

Experiments in Section 4.1 (scale equivariance). For InvarPDEs, we set hyperparameters as follows: $channel = 65, hidden\_channel = 65, iteration = 16$. This network can actually be viewed as a single PDE. On both datasets Scale-MNIST and Scale-Fashion, we train the model for 60 epochs with learning rate 2e-3 and weight decay 1e-6. For InvarLayer, we incorporate it into a CNN by replacing convolutional layers. The architecture is Layer-ReLU-MP2-BN-Layer-ReLU-MP2-BN-Layer-ReLU-MP7-BN-FC-BN-ReLU-Dr5-FC (denoting InvarLayer as "Layer", Max-Pooling with kernel size 2 as "MP2" and DropOut with rate 0.5 as "Dr5"), where the input and output dimensions of the last FC are 256 and 10, respectively. We modify the channel numbers (1-83-163-247) to keep the number of parameters almost invariant. On both datasets, we train the model for 60 epochs with learning rate 2e-3 and weight decay 1e-6.

Experiments in Section 4.2 (rotation-scale equivariance). For InvarPDEs, we set hyperparameters as follows: $channel = 25, hidden\_channel = 43, iteration = 16$. After the 4th, 8th, and 12th iterations, we apply Conv-ReLU-MP2 to double the channel numbers. This network can be viewed as four PDEs stacked together. On both datasets RS-MNIST and RS-Fashion, we report the results in Table 3 after training the model for 60 epochs with learning rate 2e-3 and weight decay 1e-6. For InvarLayer, we incorporate it into a CNN by replacing convolutional layers. The architecture is Layer-ReLU-MP2-BN-Layer-ReLU-MP2-BN-Layer-ReLU-MP7-BN-FC-BN-ReLU-Dr5-FC, where the input and output dimensions of the last FC are 256 and 10, respectively. We modify the channel numbers (1-67-132-199) to keep the number of parameters almost invariant. On both datasets, we present the results in Table 3 after training the model for 60 epochs with learning rate 8e-3 and weight decay 1e-6. If training for 90 epochs, InvarLayer lifts the accuracy (%) from 93.15 ± 0.21 to 93.40 ± 0.24 on RS-MNIST, as mentioned in the text of Section 4.2. If replacing the first MP2 with MP4 and replacing Dr5 with Dr6, InvarLayer lifts the accuracy (%) from 74.51 ± 0.71 to 76.08 ± 0.36 on RS-Fashion after training for 90 epochs.

Experiments in Section 4.3 (affine equivariance). There are two settings in this section: testing on affNIST af-

ter training on MNIST and testing on affNIST after training on affNIST. In the first setting, we use 50k non-transformed MNIST images for training and 320k affNIST images for testing. In the second setting, we use 50k affNIST images for training and 10k affNIST images for testing. In both settings, we employ the same models and the same optimization process. For InvarPDEs, we set hyperparameters as follows: $channel = 45, hidden\_channel = 45, iteration = 15$. After the 7th iteration, we apply Conv-ReLU to double the channel numbers. This network can be viewed as two PDEs stacked together. For InvarLayer, we incorporate it into ResNet-32 by directly replacing all convolutional layers without modifying the channel numbers. We train both models for 90 epochs with learning rate 1e-3 and weight decay 1e-6.